Comparative Architectures

A naïve programmer writes the following code for performing the matrix multiply-add function $C = AB + C$ on square matrices:

```c
for (i=0;i<N;++i) {
    for(j=0;j<N;++j) {
        for(k=0;k<N;++k) {
            C[k][i] = C[k][i] + ( A[k][j] * B[j][i] );
        }
    }
}
```

(where $X[v][u]$ refers to the element in row $v$, column $u$. Arrays are stored in memory row by row, i.e.
$X[0][0], X[0][1], X[0][2] \ldots X[0][N], X[1][0], X[1][1], \text{etc.}$)

(a) When used to multiply very large matrices, performance of the programmer’s algorithm is very poor. Explain what is happening. [6 marks]

(b) The algorithm can be improved simply by changing the order of the loops. Demonstrate how and why. [5 marks]

(c) Show how further improvement can be obtained through a technique known as cache blocking. [5 marks]

(d) Could the algorithm be successfully parallelised to run on a microprocessor supporting Simultaneous Multithreading (SMT)? Briefly justify your answer. [4 marks]